**Software Requirements Specification**

**for**

**ParkProximity**

**Version 1.2 approved**

**Prepared by**

| **Ooi Chee Han** | **U2222143G** |
| --- | --- |
| **Ang Ming Ze, Aaron** | **U2122736A** |
| **Arjun Kumaresh** | **U2221225J** |
| **Choh Lit Han Owen** | **U2221094K** |
| **Joel Chng Ming Xuan** | **U2121344D** |
| **Michael Santoso** | **U2221525L** |
| **Parashar Pranav** | **U2222548E** |

**Syntax Symphony**

**8 September 2023**

**Table of Contents**

[**1. Introduction 1**](#_heading=h.1fob9te)

[1.1 Purpose 1](#_heading=h.3znysh7)

[1.2 Document Conventions 1](#_heading=)

[1.3 Intended Audience and Reading Suggestions 2](#_heading=)

[1.4 Product Scope 3](#_heading=)

[1.5 References 3](#_heading=)

[**2. Overall Description 3**](#_heading=)

[2.1 Product Perspective 3](#_heading=h.2s8eyo1)

[2.2 Product Functions 4](#_heading=)

[2.3 User Classes and Characteristics 4](#_heading=)

[2.4 Operating Environment 4](#_heading=)

[2.5 Design and Implementation Constraints 4](#_heading=)

[2.6 User Documentation 4](#_heading=)

[2.7 Assumptions and Dependencies 5](#_heading=)

[**3. External Interface Requirements 5**](#_heading=)

[3.1 User Interfaces 5](#_heading=h.2jxsxqh)

[3.2 Hardware Interfaces 5](#_heading=)

[3.3 Software Interfaces 5](#_heading=)

[3.4 Communications Interfaces 5](#_heading=)

[**4. System Features 6**](#_heading=)

[4.1 Location Services 6](#_heading=)

[4.2 Carpark Data 7](#_heading=)

[4.3 Search and Filter 7](#_heading=h.8r2c4iocdm)

[4.4 Map View 7](#_heading=h.tkqb13jvdv9g)

[4.5 User Specific Features 8](#_heading=h.4vuyf6p88lsv)

[**5. Other Nonfunctional Requirements 8**](#_heading=h.3whwml4)

[5.1 Performance Requirements 8](#_heading=h.2bn6wsx)

[5.2 Safety Requirements 8](#_heading=)

[5.3 Security Requirements 8](#_heading=)

[5.4 Software Quality Attributes 9](#_heading=)

[5.5 Business Rules 9](#_heading=)

[**6. Other Requirements 10**](#_heading=)

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Michael, Owen, Joel | 14/09/23 | Moving Lab 1 Deliverables to SRS template | 1.0 |
| Chee Han | 16/09/23 | Adding non-functional requirements | 1.1 |
| Chee Han, Michael, Aaron | 21/09/23 | Adding functional requirements, use case diagram | 1.2 |

# Introduction

## Purpose

*<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>*

The product is a mobile application which displays the availability of carparks near the user and provides directions to the selected carpark. Several other features include pinpoint the parked car locations, history of carparks used, and favorite carparks spots.

## Document Conventions

*<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>*

This Software Requirement Specification (SRS) follows the following conventions:

1. Font and Typography:
   1. Text is presented in Arial font, size 11.
   2. Headings are in bold.
2. Text Formatting:
   1. Italic text is used for citations and references.
3. Highlighting:
   1. Critical requirements are highlighted in bold and red.
4. Priority Notation:
   1. Priorities for detailed requirements are inherited from their higher-level requirements unless otherwise specified.
5. Numbering or Labeling:
   1. Requirements are labeled using a unique identifier (e.g., REQ-001).
   2. Sections are numbered in the format Section X.X.
6. Terminology
   1. Specific terminology is defined in Appendix A (Glossary).
7. References:
   1. External references are cited following the APA citation style in Section 1.5.
8. Revision History:
   1. This document follows version 1.0 and was last revised on [Date]
   2. Changes to this document are tracked in the revision history section.
9. Additional Notes:
   1. Tables, diagrams, and charts are labeled and formatted following the IEEE standards.

## Intended Audience and Reading Suggestions

*<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>*

This Software Requirements Specification (SRS) is intended for various stakeholders involved in the project. The document is structured to cater to the specific needs of the following audience types:

1. **Developers**:
   1. Developers will primarily refer to this SRS to gain a high-level understanding of the project's requirements, including functional and technical specifications
2. **Project Managers:**
   1. Project Managers will use this document to assess project scope, timeline, and resource allocation. They should start with the Project Overview and Scope sections.
3. **Marketing Staff:**
   1. Marketing staff may find value in understanding the key features and user experience described in this document. They are encouraged to focus on the User Requirements and User Interface sections.
4. **Users**:
   1. End-users will benefit from a basic understanding of the system's capabilities and limitations. They should begin with the Introduction and User Requirements sections.
5. **Testers**
   1. Testers will rely on this document to create test cases and validate the system against requirements. They should start with the Functional Requirements and Testability sections.
6. **Documentation Writers**:
   1. Documentation writers should carefully review the User Documentation Requirements section to gather information for user manuals and help documentation.

**Document Contents:**

This SRS contains information related to project scope, functional and non-functional requirements, system constraints, assumptions, and other critical project details.

**Organizational Overview:**

The document is organized into the following sections: Introduction, Project Overview, Scope, Requirements, Constraints, Assumptions, and Appendices.

**Suggested Sequence for Reading:**

* Start with the Introduction to get an overall understanding.
* For a broad view of the project, proceed to the Project Overview and Scope sections.
* Developers should delve into the Requirements section, while testers can focus on the Functional Requirements and Testability sections.
* Marketing staff and users may find the User Requirements and User Interface sections most informative.
* Project Managers can gain insights from the Project Overview and Constraints sections.
* Documentation writers should thoroughly review the User Documentation Requirements section for content creation.

Tailor your reading sequence based on your specific role and objectives to extract the most relevant information from this document.

## Product Scope

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>*

The app will make use of government and public APIs to achieve the purpose laid out in 1.1.

The APIs are provided by data.gov.sg, Land Transport Authority and Google Maps.

The software framework we will use to develop the mobile app is React Native.

React Native is chosen as it allows us to easily publish the app to Android and IOS users.

The database used for the project is in SQL. In addition, the backend of the project uses Python to interact with the APIs.

## References

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

* + 1. APIs
       1. Carpark APIs
          1. [API from data.gov.sg](https://beta.data.gov.sg/datasets/85/view)
          2. [Land Transport Authority API](https://datamall.lta.gov.sg/content/datamall/en/dynamic-data.html)
          3. [Google Map API](https://developers.google.com/maps/documentation)
    2. [React Native Framework](https://reactnative.dev/)
    3. [SQLite Database](https://www.sqlite.org/docs.html)
    4. [Python Backend](https://docs.python.org/3/)

# Overall Description

## Product Perspective

*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>*

This product is to help users make use of the publicly available datasets from data.gov.sg and Land Transport Authority APIs. The product adds on to these datasets by providing search functionality and provides directions using the Google Maps API.

## Product Functions

*<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>*

* + 1. Search and filter function for users to search for carparks using their current location or manual input and filter according to distance, remaining lots and carpark price.
    2. The product must allow users to obtain directions from current location to the selected carpark.
    3. The product must allow users to enter details to save their car location to help them find their car.
    4. The product must have a favorite function to save the selected carpark for them to access easily later.

## User Classes and Characteristics

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>*

## Operating Environment

*<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>*

## Design and Implementation Constraints

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>*

## User Documentation

*<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>*

## Assumptions and Dependencies

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>*

# External Interface Requirements

## User Interfaces

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

## Hardware Interfaces

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

## Software Interfaces

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

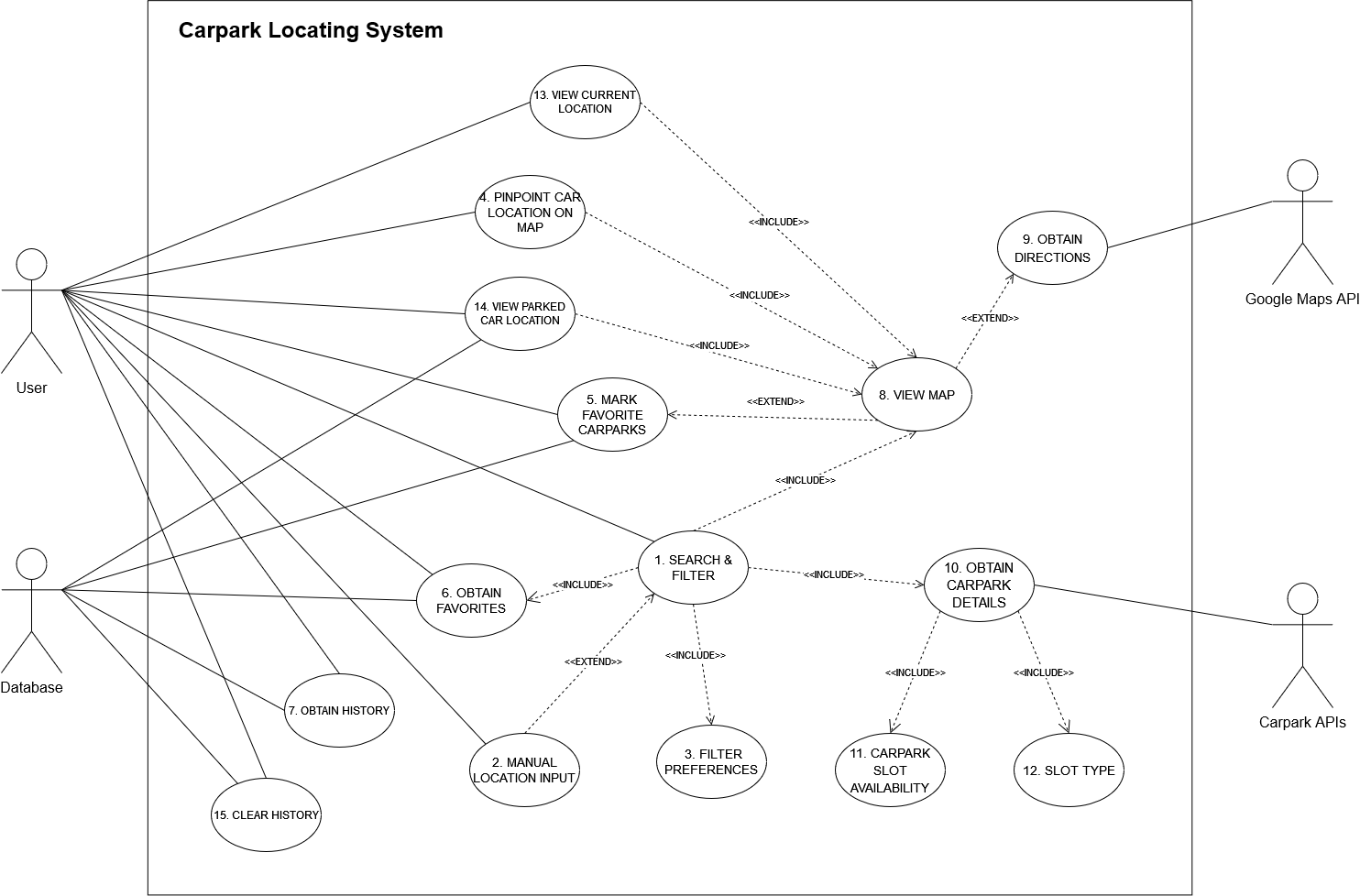
## Communications Interfaces

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

# System Features

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

Below is our user case diagram of the carpark application:

**

## Search & Filter

*<Don’t really say “System Feature 1.” State the feature name in just a few words.>*

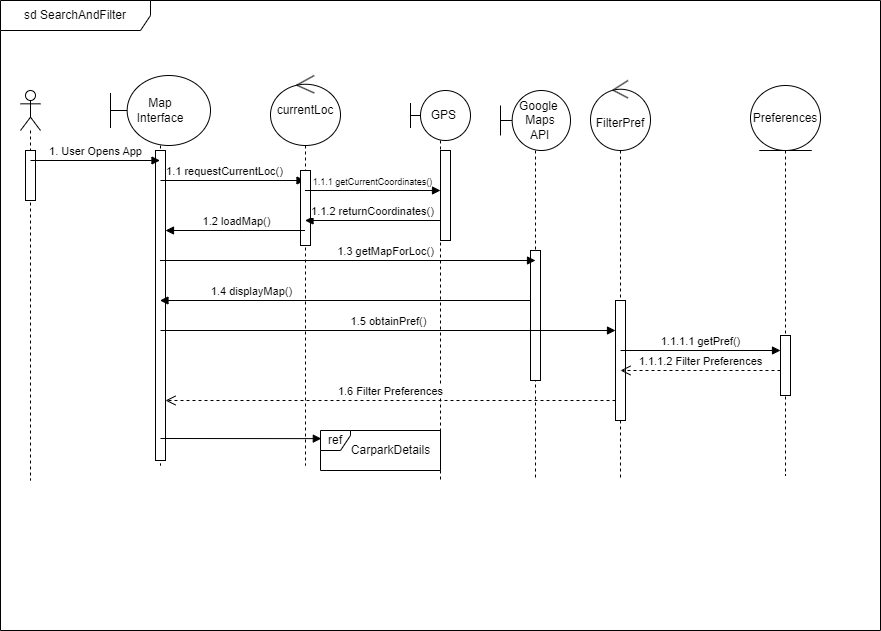
4.1.1 Description and Priority *(User case description insert here)*

*<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>*

The use case involves searching for carparks based on location, distance and availability of car park slots. This feature has a high priority.

4.1.2 Stimulus/Response Sequences *(Sequence diagram insert here)*

*<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>*

**

4.1.3 Functional Requirements *(Functional requirement insert here)*

REQ-1: Users must be able to search for carparks based on location, distance and availability

REQ-2: Users should be able to set a preferred search radius (e.g. 1 km, 5 km) for nearby carparks

REQ-3: The app must filter and display carparks that are located within the user's chosen radius

REQ-4: Carparks outside selected radius must be excluded from the results

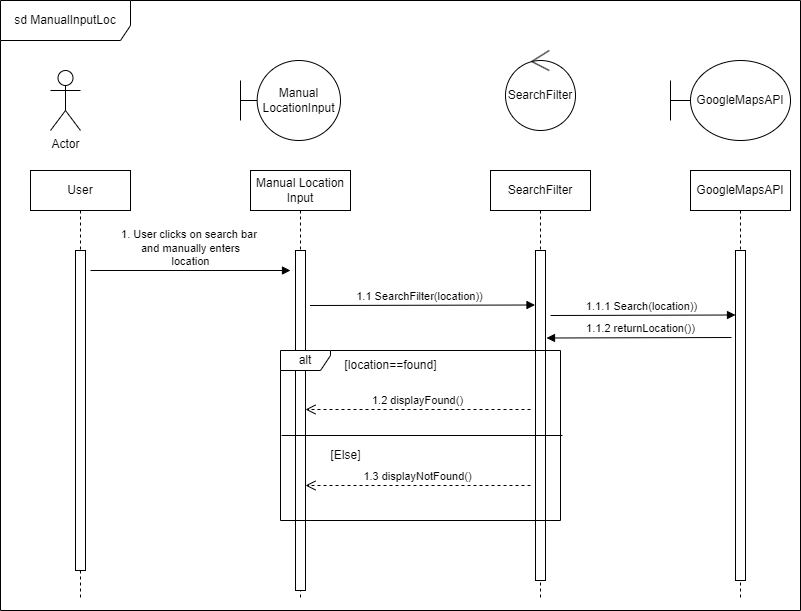
REQ-5: Carparks with no more capacity can also be excluded

## Manual Location Input

* + 1. Description and Priority

This use case allows the user to manually input a location either through an address, postal code or name of a specific car park. This feature has a medium priority.

* + 1. Stimulus/Response Sequence

**

* + 1. Functional Requirements

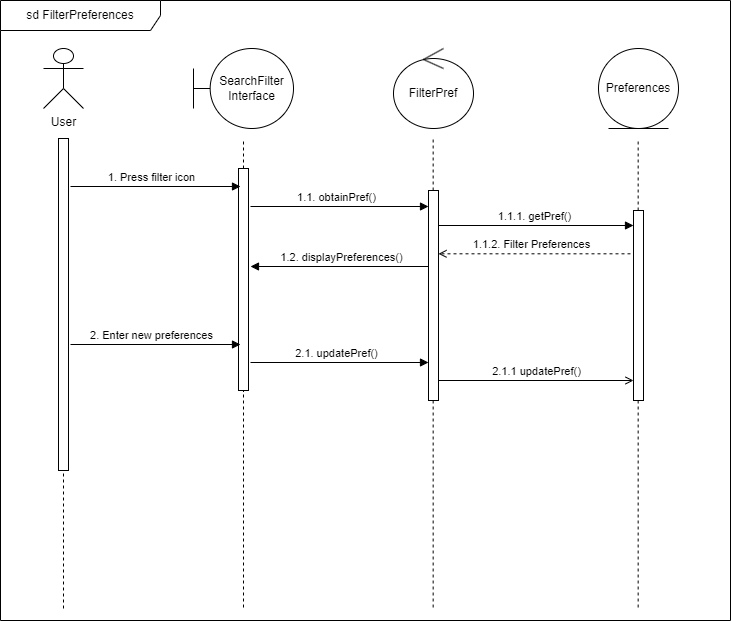
REQ-1: Carpark data must include names, addresses, capacity, available slots, pricing, distance from users and opening hours

## Filter Preferences

* + 1. Description and Priority

Users can set a preferred search radius (e.g. 1km, 5km, etc) for nearby carparks and in terms of the carpark’s slots availability. This feature has a medium priority.

* + 1. Stimulus/Response Sequence



* + 1. Functional Requirements

REQ-1: Users must be able to set a preferred search radius for nearby carparks from a range of options given to them in a drop-down selection.

## Pinpoint Car Location on Map

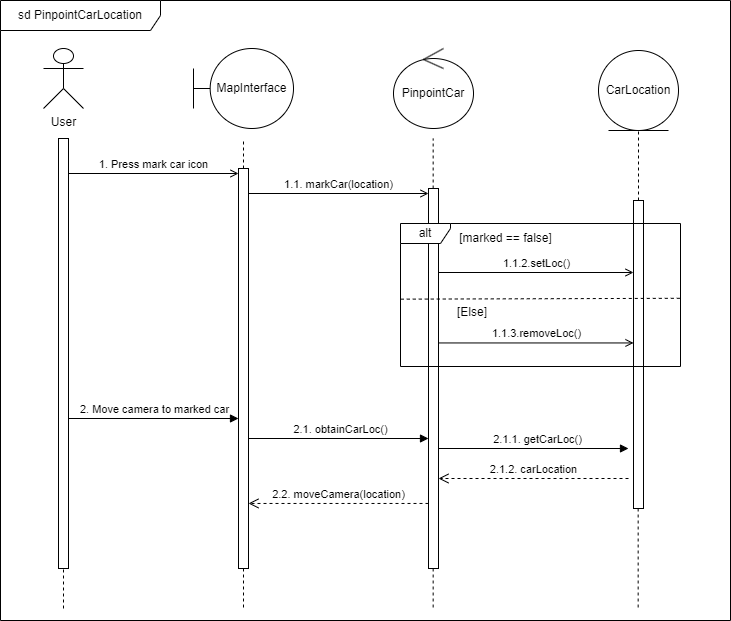
* + 1. Description and Priority

Users can pinpoint their car’s location onto the map or unmark their car’s location onto the

map after leaving the car park.

This feature has a low priority.

* + 1. Stimulus/Response Sequence



* + 1. Functional Requirements

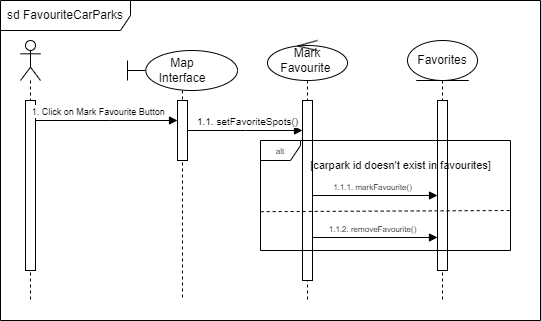
REQ-1: ParkProximity must provide pinpoint features so that users can track the car location in the map.

## Mark Favourite Carparks

* + 1. Description and Priority

Users can bookmark their handpicked car parks for future usage. This feature has a low priority.

* + 1. Stimulus/Response Sequence



* + 1. Functional Requirements

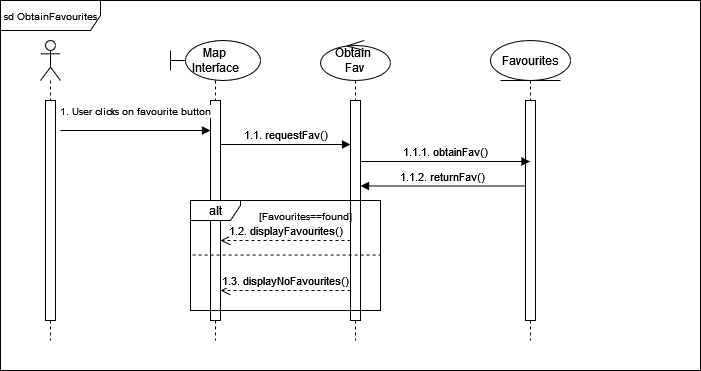
REQ-1: ParkProximity must provide an interface for a user to identify their favourite carpark(s) within the search radius revolving around their current location.

## Obtain Favourites

* + 1. Description and Priority

User can view their favourites either in the form of a list or within the information displayed after clicking on the bubble. This feature has a low priority.

* + 1. Stimulus/Response Sequence



* + 1. Functional Requirements

REQ-1: ParkProximity must provide favourite tabs to view a list of user’s handpicked car parks.

## History Management

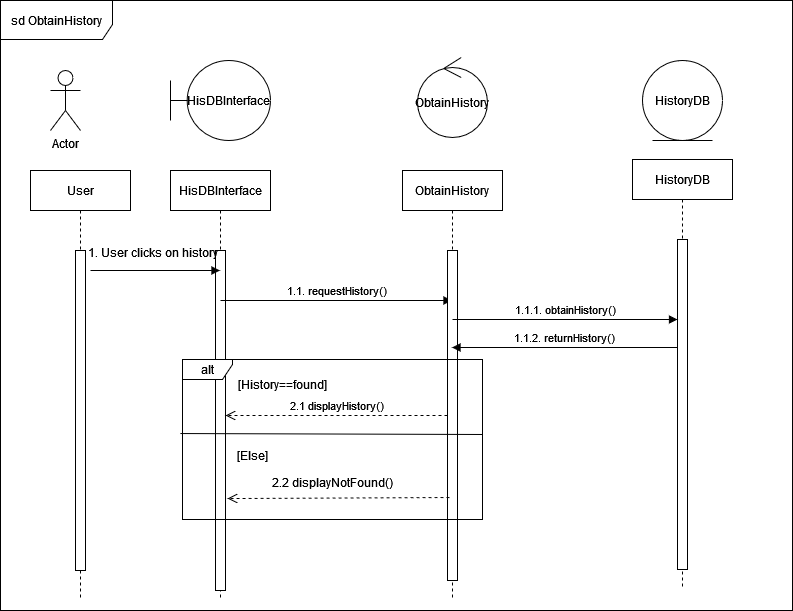
* + 1. Description and Priority

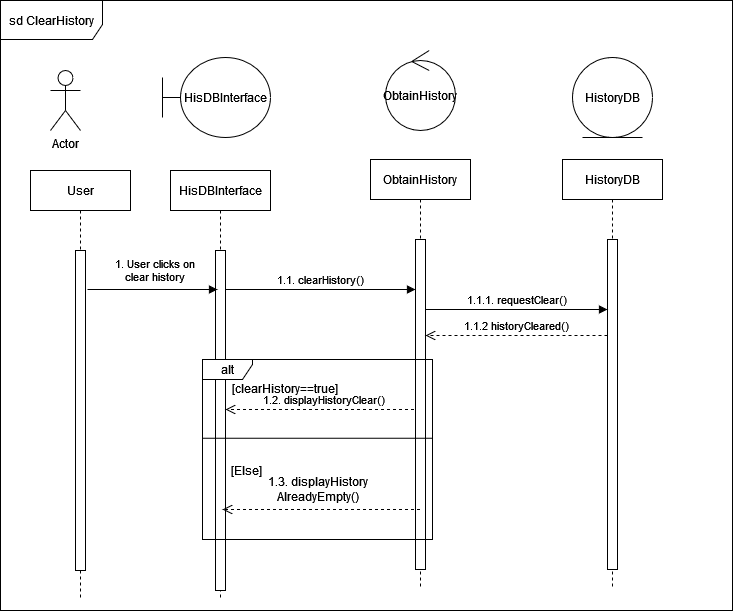
Users can view their carpark parking transactional history inclusive of carpark location details and the cost incurred from their car parking.

Users can also clear their transactional history after accumulating many records.

As this feature is not the main objective of our target users, users would not be referring to their transactional history often. Therefore, this feature would be of low priority.

* + 1. Stimulus/Response Sequence





* + 1. Functional Requirements

REQ-1: ParkProximity must have cost-tracking functions to allow users to keep track of their carpark spendings for each parking transaction.

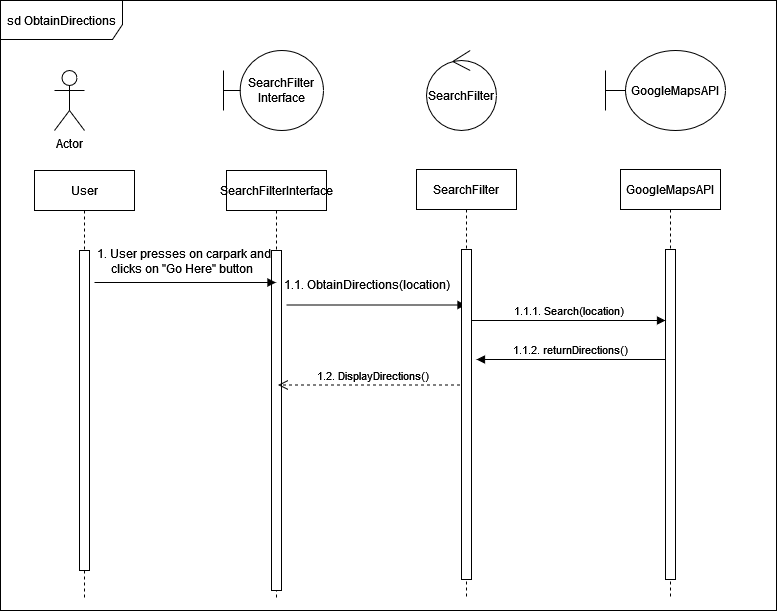
REQ-2: ParkProximity must allow users to view a list of car-parking transactions done by the users.

## Obtain Directions

* + 1. Description and Priority

After the user has chosen the specific carpark, the Google Maps API will be utilised to obtain directions to that carpark from the user’s current location. This feature has a medium priority.

* + 1. Stimulus/Response Sequence



* + 1. Functional Requirements

REQ-1: Users must have the ability to obtain directions to a selected carpark from their current location using the map.

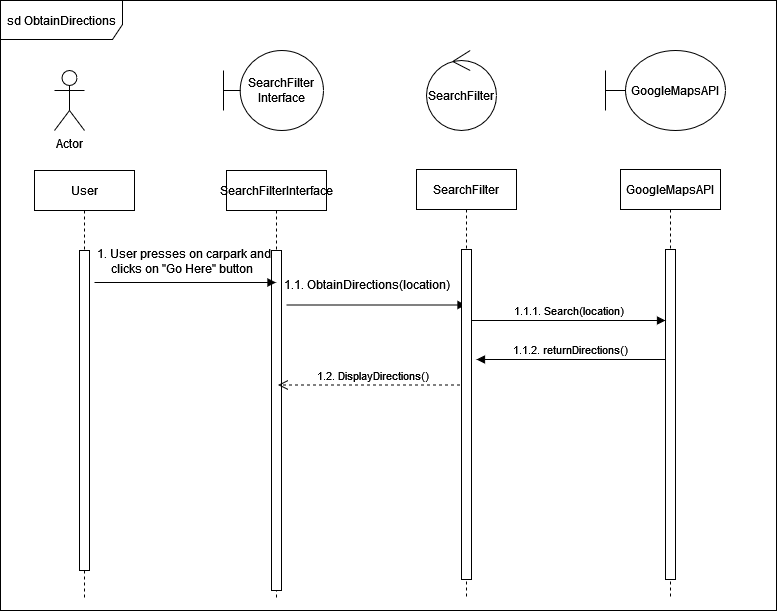
REQ-2: Directions obtained must also indicate the distance and time remaining to reach the selected carpark from the user’s current location.

## Obtain Carpark Details

* + 1. Description and Priority

The carpark details (slots availability, slot types available, total number of car park slots within a particular carpark) are retrieved from the carpark API. Such details can be represented by the colours of bubbles in the map view of . This feature has a high priority due to it being an essential requirement of ParkProximity.

* + 1. Stimulus/Response Sequence



* + 1. Functional Requirements

REQ-1: Carparks’ available slots are represented in terms of coloured bubbles:

* Green indicates the carpark is slightly occupied or fully available.
* Yellow indicates that the carpark is more than half occupied.
* Red indicates that the carpark is fully occupied.

REQ-2: Tapping the bubble (map marker) will show more information about a specific carpark inclusive of carpark name, address, price, capacity, available slots, distance from user and its opening hours.

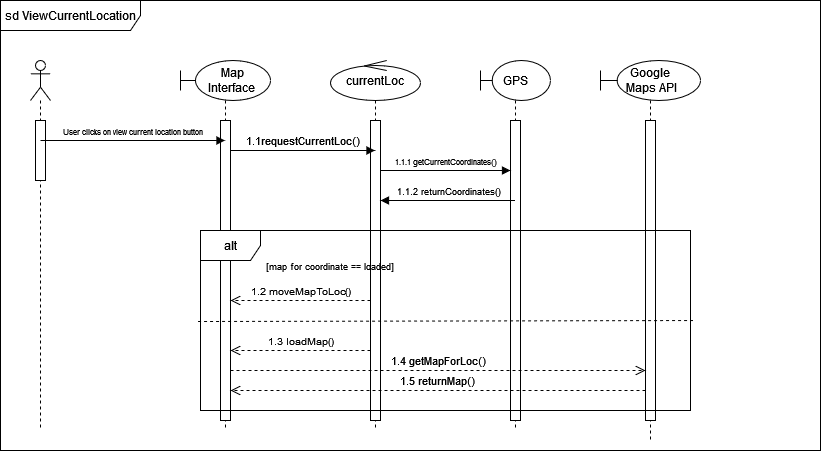
## View Current Location

* + 1. Description and Priority

Upon clicking on the current location button, the map displays the area in the perspective of the user’s current location. A blue pin indicates the user’s current location in the map view.

This feature has a high priority since the user needs to refer his/her current location with respect to other carparks of interest to the user.

* + 1. Stimulus/Response Sequence



* + 1. Functional Requirements

REQ-1: The user must be able to view his/her current location despite moving the point of interest to a place/carpark far from his/her current location.

## User Specific Features

* + 1. Description and Priority
    2. Stimulus/Response Sequence
    3. Functional Requirements

REQ-1: The app provides favourites tabs to view user’s handpicked carparks

REQ-2: The app provides pinpoint features so that user can track the car location in the map

REQ-3: The app has cost tracking functions to keep track of spendings on carpark costs

# Other Nonfunctional Requirements

## Performance Requirements

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

* + 1. App must load within 3 seconds
    2. App must respond to user actions with 1 second
    3. Carpark details updates must occur every 60 seconds

## Safety Requirements

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>*

* + 1. The app must clearly inform users of the purpose of collecting location data and seek their permission
       1. A disclaimer message would be given that users cannot use the application if the permission of collecting location data is denied.
    2. User must have the ability to disable location sharing at any time

## Security Requirements

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

*Application checklist*

*Authentication*

*Optimise Data caching*

## Software Quality Attributes

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

* + 1. Usability
       1. Users should be able to start using the app intuitively, without the need for extensive tutorials or guidance
       2. Users must be able to find available parking in no more than 2 steps
       3. Users must be able to adjust the search radius with a slider or numerical input
       4. In-app support must be available to address user questions and issues related to map functionality and location settings
    2. Compatibility
       1. Map view must be compatible with a variety of devices and screen sizes
    3. Localization
       1. Map markers and distance units must be presented in the user’s chosen language and regional settings
    4. Accessibility
       1. The map interface must include appropriate labels and alternative text for screen readers
       2. Users with disabilities must be able to interact with the map effectively
       3. Database needs to save user history from the past 3 months.
    5. Accuracy
       1. The car park information displayed should be accurate
       2. The pin-point marker must be small and visible enough to accurately indicate carpark location when there are multiple in the same vicinity
    6. Support
       1. Proper documentation must be provided for future developers
       2. The app must include help and feedback buttons for the users
       3. The app must provide FAQ for users that have questions

## Business Rules

*<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>*

# Other Requirements

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**Data Dictionary**

| **Term** | **Definition** |
| --- | --- |
| User | A person who uses our application |
| Carpark | An area where cars or other vehicles can be left temporarily |
| Global Positioning System (GPS) | A service that provides users with positioning, navigation, and timing (PNT) services |
| Location services | General term for software service for mobile users to identify their current physical location |
| Map view | A drawing of a particular area, showing its main features as viewed from above |
| Application Programming Interface (API) | A software service acting as an intermediary layer which returns the data requested by our application |
| Direction | A simple drawing on the map view to show direction of travel between two locations |
| Search | Returns the specified carpark or the nearest carpark from the searched address |
| Filter | Displays the search results according to the specified rule which is by search radius |
| Favourite | To record the selected address or carpark for future quick access |
| Frequently Asked Question (FAQ) | A list of questions and their answers, intended to help users understand how to use a particular function of our application |
| Search radius | A shaded circular region on the map view to indicate carparks within the specified distance from a location |

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*

Source: http://www.frontiernet.net/~kwiegers/process\_assets/srs\_template.doc